In the claims:

For the Examiner's convenience, all pending claims are presented below with changes shown in accordance with the new mandatory amendment format. Please amend claims 21-24, 27, 29-34, 36, 37, 39, 41 and 42.

1 1-20   Cancelled

1 21.   (Currently Amended) A method comprising:

2   generating a first test program population to test the functionality of an integrated

3 circuit (IC), the first test program population ~~including~~ comprising a plurality of test

4 programs, each test program having a first set of instructions and data;

5   executing each of the test programs in the first test program population;

6   evaluating a first set of coverage data from the first test program population to

7 determine if the IC has been sufficiently tested, wherein evaluating the first set of

8 coverage data comprises comparing the coverage data to a predetermined coverage

9 requirement; and

10   generating a second program population if the IC has not been sufficiently tested

11 by the first test program population, the second test program population ~~including~~

12 comprising ~~an~~ a plurality of updated test programs, wherein each updated test program

13 ~~population having a second set of instructions and data being~~ is a mutation of a test of

14 the first test program ~~original~~ population for a combination of two or more of the test

15 programs of the first test program population.

1 22.   (Currently Amended) The method of claim 21, further comprising:

2   executing the second test program population.

1 23.   (Currently Amended) The method of claim 22, wherein generating the first test

2 program population comprises:

Docket No.: 042390.P6602
Application No.: 09/475,526

3   generating a first abstract syntax tree (AST);

4   generating the first set of instructions and data for the first AST; and

5   translating the first AST into a first executable test program.

1   24.   (Currently Amended) The method of claim 23, wherein generating the second test

2   program population comprises:

3   generating a second abstract syntax tree (AST);

4   generating the a second set of instructions and data for the second AST; and

5   translating the second AST into a second executable test program population.

1   25.   (Previously Presented)   The method of claim 24, further comprising

2   mutating a selected AST.

1   26.   (Previously Presented)   The method of claim 25, wherein mutating a

2   selected AST comprises:

3   selecting an AST;

4   removing a segment of the selected AST; and

5   inserting a replacement segment into the selected AST to form a mutated AST.

1   27.   (Currently Amended) The method of claim 26, further comprising:

2   generating a third set of instructions and data for the mutated AST; and

3   translating the mutated AST into a third executable test program population.

1   28.   (Previously Presented)   The method of claim 25, wherein mutating a

2   selected AST comprises:

3   selecting the first AST and the second AST; and

4   combining a segment of the first AST with a segment of the second AST to form

5   a mutated AST.

1   29.   (Currently Amended) The method of claim 28, further comprising:

2   generating a third set of instructions and data for the mutated AST; and

3   translating the mutated AST into a third executable test program <u>population</u>.

1   30.   (Currently Amended) The method of claim 23, further comprising:

2   adding the first AST and the first set of coverage data into <u>a</u> test program

3   ~~population~~ after the first test program <u>population</u> has been executed.

1   31.   (Currently Amended) A computer system comprising:

2   a storage device coupled to a processor and having stored therein at least one

3   routine, which when executed by the processor, causes the processor to generate data, the

4   routine causing the processor to,

5   generate a first test program <u>population</u> to test the functionality of an integrated

6   circuit (IC), the first test program population ~~including~~ <u>comprising</u> a <u>plurality of</u> test

7   <u>programs, each test</u> program having a first set of instructions and data;

8   execute <u>each of the test programs in</u> the first test program <u>population</u>;

9   evaluate a first set of coverage data from the first test program <u>population</u> to

10   determine if the IC has been sufficiently tested, wherein evaluating the first set of

11   coverage data comprises comparing the coverage data to a predetermined coverage; and

12   generate a second program <u>population</u> if the IC has not been sufficiently tested by

13   the first test program <u>population</u>, the second test program <u>population</u> ~~including~~

14   <u>comprising</u> ~~an~~<u>a plurality of</u> updated test <u>programs, wherein each updated test program</u>

15   ~~population having a second set of instructions and data being~~ <u>is</u> a mutation of <u>a test of</u>

16   <u>the first test program</u>~~original~~ population <u>for a combination of two or more of the test</u>

17   <u>programs of the first test program population</u>.

1   32.   (Currently Amended) The computer system of claim 31, wherein the routine

2   further causes the processor to,

3   execute the second test program <u>population</u>.

1   33.   (Currently Amended) The computer system of claim 32, wherein generating the

2   first test program population comprises:

3   generating a first abstract syntax tree (AST);

4   generating the first set of instructions and data for the first AST; and

5   translating the first AST into a first executable test program.

1   34.   (Currently Amended) The computer system of claim 33, wherein generating the

2   second test program population comprises:

3   generating a second abstract syntax tree (AST);

4   generating the a second set of instructions and data for the second AST; and

5   translating the second AST into a second executable test program population.

1   35.   (Previously Presented)   The computer system of claim 34, wherein the

2   routine further causes the processor to mutate a selected AST.

1   36.   (Previously Presented)   The computer system of claim 35, wherein mutating

2   a selected AST comprises:

3   selecting an AST;

4   removing a segment of the selected AST; and

5   inserting a replacement segment into the selected AST to form a mutated AST.

1   37.   (Currently Amended) The computer system of claim 36, wherein the routine

2   further causes the processor to,

3   generate a third set of instructions and data for the mutated AST; and

4   translate the mutated AST into a third executable test program population.

1   38.   (Previously Presented)   The computer system of claim 35, wherein mutating

2   a selected AST comprises:

3   selecting the first AST and the second AST; and

4   combining a segment of the first AST with a segment of the second AST to form

5   a mutated AST.

1   39.   (Currently Amended) The computer system of claim 38, wherein the routine

2   further causes the processor to,

3   generating a third set of instructions and data for the mutated AST; and

4   translating the mutated AST into a third executable test program <u>population</u>.

1   40.   (Previously Presented)   The computer system of claim 33, wherein the

2   routine further causes the processor to,

3   add the first AST the first set of coverage data into test program population after

4   the first test program has been executed.

1   41.   (Currently Amended) A validation test system comprising:

2   a test builder to generate test <u>program populations</u> ~~programs~~ to test the

3   functionality of an integrated circuit (IC);

4   a test generator to translate the test <u>program populations</u> ~~programs~~ into an

5   executable test;

6   a test analyzer to execute the test <u>program populations</u> ~~programs~~; and

7   a feedback engine to build and update a population of test programs by generating

8   an abstract syntax tree (AST) for each test <u>program populations</u> ~~programs~~.

1   42.   (Currently Amended) The system of claim 41, wherein the feedback engine

2   determines whether a predetermined test program population threshold has been reached

3   after a test <u>program populations</u> ~~programs~~ has been executed.

1   43.   (Previously Presented)   The system of claim 42, wherein the feedback

2   engine generates one or more mutated ASTs if it is determined that the predetermined test

3   program population threshold has been reached.

Docket No.: 042390.P6602
Application No.: 09/475,526
                                    6

1   44.   (Previously Presented)   The system of claim 43, wherein the feedback

2   engine generates a mutated AST by selecting a first AST, removing a segment of the first

3   AST and inserting a replacement segment into the first AST.

1   45.   (Previously Presented)   The system of claim 43, wherein the feedback

2   engine generates a mutated AST by selecting a first AST and a second AST and

3   combining a segment of the first AST with a segment of the second AST to form.

1